PERGAMON Computers and Mathematics with Applications 44 (2002) 125–141

An International Journal
computers & mathematics with applications

www.elsevier.com/locate/camwa

Numerical Experiments of Some Krylov Subspace Methods for Black Oil Model

JIANWEN CAO*
R & D Center for Parallel Software
Institute of Software
Chinese Academy of Sciences
Beijing 8718, 100080, P.R. China
cao@mail.rdcps.ac.cn

CHOI-HONG LAI
School of Computing and Mathematical Sciences
University of Greenwich
30 Park Row, Greenwich, London SE10 9LS, U.K.
C.H.Lai@gre.ac.uk

(Received and accepted July 2001)

Abstract—This paper discusses preconditioned Krylov subspace methods for solving large scale linear systems that originate from oil reservoir numerical simulations. Two types of preconditioners, one being based on an incomplete LU decomposition and the other being based on iterative algorithms, are used together in a combination strategy in order to achieve an adaptive and efficient preconditioner. Numerical tests show that different Krylov subspace methods combining with appropriate preconditioners are able to achieve optimal performance.

Keywords—Krylov subspace methods, Black oil model, Large scale linear system, Preconditioner, Effective linear solver.

1. INTRODUCTION

Many preconditioned Krylov subspace methods [1] which have been proposed for the solution of large scale linear systems, theoretical analyses, and numerical experiments may be found in the literature. In this paper, we consider a specific type of linear system originating from the black oil model in oil reservoir simulation. There exist some Krylov subspace algorithms and preconditioning techniques for the black oil model as have appeared in the literature such as magazines of the Society of Petroleum Engineers (SPE). However, comparison between different algorithms with different choices of preconditioners have not been presented in a systematic way for the present industrial problems. The emphasis of this paper is on the numerical experimentation and comparison of some Krylov subspace algorithms using different preconditioners.

*This author is partially supported by the Large Scale Scientific Computation Research Project (973) of China and a visiting fellowship at the School of Computing and Mathematical Sciences, University of Greenwich.

0898-1221/02/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0898-1221(02)00135-9

Krylov subspace methods are often being used for solving large scale linear systems. These methods are essentially based on orthogonalisation. Early versions were initially developed by a number of authors including Lanczos [2]. The motivation behind the Lanczos method came from eigenvalue problems, and the method was based on two mutually orthogonal vector sequences. Such an approach was later applied to solve symmetric linear systems. An independent discovery of the same method was given by Hestenes and Stiefel [3], in which the conjugate gradient (CG) method for solving linear systems was proposed and presented as a direct method. The two-term recurrences in the Hestenes/Stiefel method may be combined to eliminate the search directions, which gives the Lanczos method as discussed in Arnold's paper [4]. On the other hand, the Arnoldi algorithm may be applied to nonsymmetric systems. Methods developed as such have not received much attention in solving systems of linear equations until [5]. In later years, various methods based on orthogonal properties for nonsymmetric systems occur. Some characterisation work, including [6–8], and some good review work, including [1,9,10], provides a good insight into techniques that are based on the projection process onto a Krylov subspace.

Three typical types of Krylov subspace methods are described below. A linear system $Ax = f$ satisfies the condition

$$f - Ax^{(m)} \perp \mathcal{L}_m,$$

where $\mathcal{L}_m$ is a subspace of dimension m and has a projection method which generates approximate solutions $x^{(m)}$ in the affine shifted Krylov subspace

$$x^{(0)} + \mathcal{K}_m\left(A, r^{(0)}\right) = x^{(0)} + \operatorname{span}\left\{r^{(0)}, Ar^{(0)}, \ldots, A^{m-1}r^{(0)}\right\}$$

of dimension m where $r^{(0)} = f - Ax^{(0)}$ and $x^{(0)}$ is an initial approximation. There are three primary choices of the subspace $\mathcal{L}_m$, each of which generates a different class of Krylov subspace projection methods.

First,

$$\mathcal{L}_m = \mathcal{K}_m\left(A, r^{(0)}\right),$$

which leads to many popular methods such as CG, Lanczos method, and FOM [1]. Second,

$$\mathcal{L}_m = A\mathcal{K}_m\left(A, r^{(0)}\right),$$

which leads to methods like GMRES and Orthomin. Third,

$$\mathcal{L}_m = \mathcal{K}_m\left(A^{\top}, r^{(0)}\right),$$

which leads to several bi-orthogonal algorithms such as BiCG, BiCGSTAB, CGS, QMR, GPBi-CG, BiCGSTAB(ell), and ML(k)BiCGSTAB. In particular, the CG, which belongs to the first type, is the oldest known Krylov subspace method and is suitable for symmetric positive definite systems, and requires us to minimize the A-norm of the error, $\|e^{(m+1)}\|_A \equiv \langle e^{(m+1)}, Ae^{(m+1)}\rangle^{1/2}$, where $e^{(m+1)} = A^{-1}f - x^{(m+1)}$ is the error. The method requires generating vector sequences of approximation $\{x^{(m)}\}$, the corresponding residual $\{r^{(m)}\}$, and search directions $\{p^{(m)}\}$ to be used in the update of iterative approximations and residuals. An implementation of the algorithm is listed below for references within this article.

Algorithm:- CG

Input $x^{(0)}$; $r^{(0)} := f - Ax^{(0)}$, $p^{(0)} := r^{(0)}$, $m := 0$;

For $m = 1, 2, \ldots$

Compute $Ap^{(m-1)}$;

$\alpha := \langle r^{(m-1)}, r^{(m-1)}\rangle / \langle p^{(m-1)}, Ap^{(m-1)}\rangle$;

$x^{(m)} := x^{(m-1)} + \alpha p^{(m-1)}$;

$r^{(m)} := r^{(m-1)} - \alpha A p^{(m-1)}$;
$\beta := \langle r^{(m)}, r^{(m)} \rangle / \langle r^{(m-1)}, r^{(m-1)} \rangle$;
$p^{(m)} := r^{(m)} + \beta p^{(m-1)}$;
End-For;
End-Algorithm

The convergence rate of Krylov subspace methods depends mainly on the distribution of eigenvalues of the matrices. If these eigenvalues are clustered (e.g., around 1), the L_2-norm of residual vector $r^{(m)}$ from any of the above Krylov subspace methods may converge to zero in a fast and steady fashion. One method to produce clustered eigenvalues is to use *good* preconditioners.

For a given preconditioner, different implementations lead to the same set of eigenvalues of the preconditioned matrix. However, convergence behaviours may be different for different implementations—the reason being that these implementations have different sets of eigenvectors or, more specifically, different implementations give different components of the initial residual ($r^{(0)}$) along the corresponding eigenvector direction. The idea is easily demonstrated here. Suppose B is a preconditioner; the left preconditioning results in the equation

$$BAx = Bf, \tag{1}$$

while the right preconditioning results in the equation

$$ABB^{-1}x = f, \tag{2}$$

and solutions to these equations are required. Assuming that the GMRES method is being used, equation (1) requires the minimisation of the preconditioned residual $B(f - Ax^{(m)})$, while (2) requires the minimisation of the original residual $f - Ax^{(m)}$. The two minimisation strategies result in very different convergence behaviour. First, the initial search direction vectors of the two minimisation strategies are $B(f - Ax^{(0)})$ and $f - Ax^{(0)}$, respectively. The corresponding two spanned Krylov subspaces are

$$Br^{(0)}, BABr^{(0)}, (BA)^2 Br^{(0)}, \ldots, (BA)^{m-1} Br^{(0)}$$

and

$$r^{(0)}, Ar^{(0)}, A^2 r^{(0)}, \ldots, A^{m-1} r^{(0)}.$$

Each solution lies in the Krylov subspace that is calculated by using an orthogonal basis of the subspace. Hence, different Krylov subspaces alter the convergence rates. Second, the left preconditioning allows the construction of an orthogonal basis for the above left preconditioned Krylov subspace; all residual vectors and their norms correspond to the preconditioned residuals $B(f - Ax^{(m)})$, while the original residuals $f - Ax^{(m)}$ are not easy to obtain unless they are computed explicitly. This may lead to some difficulties if the stopping criterion is desired to check the actual residuals rather than those preconditioned ones. However, the right preconditioning can easily allow the construction of an orthogonal basis based on the actual residuals, and the stopping criterion can easily be checked by means of the actual residuals. An implementation of the preconditioned CG algorithm is listed below for references within this article.

Algorithm:- Preconditioned CG
Input $x^{(0)}$; $r^{(0)} := f - Ax^{(0)}$;
Solve $Mv^{(0)} = r^{(0)}$; $p^{(0)} := v^{(0)}$; $m := 0$;
For $m = 1, 2, \ldots$
Compute $Ap^{(m-1)}$;
$\alpha := \langle r^{(m-1)}, v^{(m-1)} \rangle / \langle p^{(m-1)}, Ap^{(m-1)} \rangle$;
$x^{(m)} := x^{(m-1)} + \alpha p^{(m-1)}$;

$r^{(m)} := r^{(m-1)} - \alpha A p^{(m-1)}$;
Solve $Mv^{(m)} = r^{(m)}$;
$\beta := \langle r^{(m)}, v^{(m)} \rangle / \langle r^{(m-1)}, v^{(m-1)} \rangle$;
$p^{(m)} := v^{(m)} + \beta p^{(m-1)}$;
End-For;
End-Algorithm

The organization of the paper is as follows. In Section 2, an overview is given of the linear system arised from the black oil model in oil reservoir simulation. In Section 3, a discussion is given of the algorithms and the preconditioners for solving the linear system. In Section 4, comparison of different algorithms and different preconditioners are included. Some concluding remarks are given in Section 5.

2. THE BLACK OIL MODEL

The black oil model is an important mathematical model used in oil reservoir numerical simulation. In this model, there are three distinct phases, namely, oil, water, and gas. Fluids of different phases are usually considered to be at constant temperature and in thermodynamic equilibrium throughout the entire reservoir in consideration. In order to construct sets of simultaneous partial differential equations for flow in reservoirs, Darcy's law is adopted to provide a relationship [11] between the pressure gradient ∇P_l and its flow rate $\vec{U_l}$, where $l = o$, g, and w denote oil, gas, and water, respectively, for multiphase flow in oil reservoir simulation,

$$\vec{U_l} = -\frac{kk_{rl}}{\mu_l}(\nabla P_l - \gamma \nabla z).$$

Considering mass conservation law being applied to the domain of interest with grid cells for oil, water, and gas phase, leads to the following set of coupled time-dependent partial differential equations

$$\begin{aligned}
\nabla \cdot (K_l \nabla (P_l - \gamma_l z)) + Q_l + \lambda_l \{\nabla \cdot ((K_o R_s) \nabla \Phi_o) + Q_o R_s\} &= \frac{\partial}{\partial t}\{\phi b_l S_l + \lambda_l (\phi b_o S_o R_s)\}, \\
l &= o, g, w, \\
\lambda_l &\equiv 1 \text{ (when } l = g), \ 0 \text{ (when } l = o, w), \\
K_l &= \frac{kk_{rl}b_l}{\mu_l} = f_1(P_o, S_w, S_x), \\
k_{rw} &= f_w(S_w), \\
k_{ro} &= f_o(S_w, S_g), \\
k_{rg} &= f_g(S_g), \\
b_w &= f_2(P_o), \\
b_o &= f_3(P_o, S_x), \\
b_g &= f_4(P_o, Z, T), \\
P_w &= f_5(P_o, S_w), \\
P_g &= f_6(P_o, S_g), \\
\phi &= f_7(P_o), \\
R_s &= f_8(P_s), \\
S_o &= 1 - S_w - S_g, \\
S_x &= P_s \qquad (\text{if } S_g = 0, \ P_o > P_s), \\
&= S_g \qquad (\text{if } S_g > 0, \ P_o = P_s), \\
Q_{l,ijk} &= WC_{ijk} \times K_l \times (P_{o,ijk} - P_{wf,ijk}).
\end{aligned} \tag{3}$$

Here K_l is the transmissibility for phase l, k is the absolute permeability tensor of the porous media (to be determined experimentally), k_{rl} is the relative permeability for phase l, μ_l is the viscosity of phase l, S_l is the saturation of phase l, and Q_l is the well flow term for phase l (or the mass accumulation per unit time). It should be noted that the saturation of different phases satisfies the relation $\sum_l S_l = 1$. More details of the variables and their physical properties can be found in many literatures and related references, e.g., [11]. The model is being used in the commercial reservoir simulation software packages such as Simbest-II [12], VIP [13], and ECLIPSE [14]. The model represents mathematically a class of important industrial problems rather than simply being an idealised model for benchmark tests and uses realistic saturation coefficients, permeability, and transmissibility which are in-situ field data collected over a long period of time.

Using an implicit scheme along the temporal axis, the partial differential equations (3) become a system of nonlinear equations at each time step. Newton's method is one of the most popular choices for the linearisation of nonlinear systems, arising from the implicit discretization of the black oil model at each time step, which leads to the 3×3 block linear system

$$Ax = f$$

or

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}, \tag{4}$$

where A is the coefficient matrix, f is the vector representing the right-hand side, and x is the vector of unknowns. Here A_{11} is the coefficient submatrix obtained from pressure equation in oil phase, A_{22} is obtained from the saturation equation in water phase, and A_{33} is obtained from the saturation equation in oil (or gas) phase. The matrices A_{ij} are heptadiagonal matrices. x_1, x_2, and x_3 are the oil phase pressure ($P_{o,ijk}$), the water saturation ($S_{w,ijk}$), and the oil (or gas) saturation ($S_{x,ijk}$), respectively. Here ($ijk = 1, 2, \ldots, N_g$), N_g is the number of grid cells in the oil field simulation domain.

3. LINEAR SOLVERS FOR OIL RESERVOIR SIMULATION

Each Newton iteration leads to a nonsymmetric linear system as represented by (4), and it is usually a large system. Parallel or distributed processing is required to solve such large scale problems in order to reduce computational time. In doing so, the linear system is split across a number of processors depending on the speed of the CPU, the memory size of a computer, or, sometimes, the preference of the user. The partition of the problem is based on the size of grid, $N_g = N_x N_y N_z$ where N_x, N_y, and N_z are the number of grid points along x-, y-, and z-axis, respectively. For simplicity, the partitioning of grid cells is only performed along the x- and y-axis leaving grid cells along the z-axis to remain intact.

Several Krylov subspace methods with combined preconditioners were adopted to solve the split linear system. The goal of this paper is to examine and compare parallel solvers that can be used to deal with three-dimensional partitioned grids and, at the same time, provide a scalable and portable code across a broad range of PC clusters and workstations using either shared or distributed memory architectures. For this purpose, the preconditioned parallel linear solvers use both domain decomposition and data parallelisation techniques [15].

To implement iterative schemes on a grid which is partitioned into a number of subgrids in a domain decomposition context, the data close to the partition boundaries of each subgrid and local to a processor needs to be exchanged across its neighbouring processors. Each processor contains a subgrid surrounded by a number of duplicated grid points that are the duplication of grid points contained in the neighbouring processors. Interprocessor communications use a standard communication massage passing interface library (MPI), which is system independent.

The oil field wellpores are provided to the authors in terms of index and coefficient arrays which are stored globally across the processors. A processor locally implements those well arrays that only exist in the subgrid system. The corresponding arrays of fault links, crossflow, radial closure, and local grid refinement (LGR) are processed in a similar way as the well arrays.

The parallel linear solvers that the authors used in this paper allow users to nominate one of the following Krylov subspace methods: Orthomin [16], GMRES [8], BiCGSTAB [17,18], and GPBi-CG [19]. Each of these Krylov subspace methods is equipped with a combined preconditioning strategy.

Orthomin is a popular algorithm used in commercial oil reservoir simulation software as a linear solver for nonsymmetric linear systems. The direction vectors, $p^{(m)}$, of the update

$$x^{(m)} = x^{(0)} + \sum_{k=0}^{m-1} \alpha_k p^{(k)}$$

in Orthomin is *A-orthogonal*; i.e., the algorithm constructs its residual vector $r^{(m)}$ in such a way that the orthogonality condition

$$\left(r^{(m)}, Ap^{(k)}\right) = 0, \qquad k = 0, 1, \ldots, m-1,$$

is satisfied. Minimisation of the residual in $A\mathcal{K}_m(A, r^{(0)})$ requires the residual updates in the form $r^{(m)} = r^{(m-1)} - \alpha_{m-1} Ap^{(m-1)}$ where α_{m-1} can be easily shown to be $\langle r^{(m-1)}, Ap^{(m-1)}\rangle / \langle Ap^{(m-1)}, Ap^{(m-1)}\rangle$. To satisfy the above orthogonality condition, the projection of $r^{(m)}$ in the direction of $Ar^{(m)}$ is subtracted, but the orthogonality with $Ap^{(m-1)}$ is kept. Therefore, an implementation of Orthomin can be listed in the algorithm below.

Algorithm:- Orthomin
  Input $x^{(0)}$; $r^{(0)} := f - Ax^{(0)}$, $p^{(0)} := r^{(0)}$, $m := 0$;
  For $m = 1, 2, \ldots$
    Compute $Ap^{(m-1)}$;
    $\alpha := \langle r^{(m-1)}, Ap^{(m-1)}\rangle / \langle Ap^{(m-1)}, Ap^{(m-1)}\rangle$;
    $x^{(m)} := x^{(m-1)} + \alpha p^{(m-1)}$;
    $r^{(m)} := r^{(m-1)} - \alpha Ap^{(m-1)}$;
    $\beta := -\langle Ar^{(m)}, Ap^{(m-1)}\rangle / \langle Ap^{(m-1)}, Ap^{(m-1)}\rangle$;
    $p^{(m)} := r^{(m)} + \beta p^{(m-1)}$;
  End-For;
End-Algorithm

The direction vectors $p^{(m)}$ in GMRES is also known to be *A-orthogonal*, but GMRES constructs its residual vector $r^{(m)}$ in such a way that the L_2-norm $\|r^{(m)}\|_2$ is minimised. The approximate solution is updated in the form $x^{(m)} = x^{(0)} + Q_m y_m$ where y_m minimizes the function $\mathcal{J}(y) = \|\beta e_1 - \widetilde{H}_m y\|_2$, i.e.,

$$y_m = \min_{y \in \Re^m} \left\| \beta e_1 - \widetilde{H}_m y \right\|_2 .$$

Here Q_m is the $n \times m$ matrix with column vectors $p_1, \ldots, p_m$ and $\widetilde{H}_m$ is the $(m+1) \times m$ Hessenberg matrix with nonzero entries $h_{i,j}$ as defined in the algorithm shown below, $\beta := \|r^{(0)}\|_2$, and $e_1 = (1\ 0\ \cdots\ 0)^\top$. Minimisation of the residual in $A\mathcal{K}_m(A, r^{(0)})$ requires the solution of the above least squares problem. Therefore, an implementation of GMRES can be written as the algorithm below.

Algorithm:- GMRES
  Input $x^{(0)}$, $r^{(0)} := f - Ax^{(0)}$, $\beta := \|r^{(0)}\|_2$, $p_1 := r^{(0)}/\beta$, $m := 0$;
  For $m = 1, 2, \ldots$

$h_{i,m} := (Ap_m, p_i)$, $i = 1, 2, \ldots, m$
$\hat{p}_{m+1} := Ap_m - \sum_{i=1}^{m} h_{i,m} p_i$
$h_{m+1,m} := \|\hat{p}_{m+1}\|_2$
$p_{m+1} := \hat{p}_{m+1}/h_{m+1,m}$
$y_m :=$ least-squares solution to $\widetilde{H}_m y \approx \beta e_1$
$x^{(m)} := x^{(0)} + Q_m y_m$;
$r^{(m)} := Q_{m+1}(\beta e_1 - \widetilde{H}_m y_m)$;
End-For;
End-Algorithm

This algorithm utilizes a modified Gram-Schmidt orthogonalisation method based on the Arnoldi algorithm to construct an orthogonal basis. For details of this orthogonalisation process and the minimisation process, readers are referred to [20].

Note that the convergence of GMRES is strictly monotonic in most cases; the implementation is robust and attractive to reservoir simulation applications. In fact, many current commercial reservoir simulators use GMRES to solve the resulting large-scale linear systems. Apparently, the GMRES method requires an increasing amount of work and storage per iteration, particularly for commercial reservoir simulations involving nonsymmetric linear systems. As a result, the industry is constantly looking at alternatives for very large simulations. On the other hand, it is possible to use GMRES(m) algorithm, which involves a restart of GMRES method every m steps. The initial guess at each GMRES restart is the latest iterative approximation of the last restart. The restarted GMRES would provide a faster convergence rate and save much memory cost as confirmed by the numerical tests shown in Section 4.

BiCG-type algorithm attempts to construct a suitable set of basis vectors in a Krylov subspace by means of the three-term recurrence relation

$$\alpha_{j+1} r^{(j+1)} = Ar^{(j)} - \beta_j r^{(j)} - \gamma_j r^{(j+1)}.$$

In the symmetric case, it is easy to determine these coefficients so that the basis vectors are orthogonal. However, it is not possible to do so in the nonsymmetric case. In a Lanczos bi-orthogonalisation process, it is possible to recursively generate the basis vector $\{r^{(0)}, r^{(1)}, \ldots, r^{(j-1)}\}$ in the Krylov subspace $\mathcal{K}_j(A, r^{(0)})$. Let $r^{(j)}$ result from the above three-term recurrence and satisfy the following relation:

$$r^{(j)} \perp \mathcal{K}_j\left(A^{\top}, \hat{r}^{(0)}\right).$$

Here $\hat{r}^{(0)}$ is an arbitrary vector. Similarly, let $\hat{r}^{(j)}$ satisfy a similar relation as that for $r^{(j)}$. Therefore, it leads to two sets of vectors, namely $\{\hat{r}^{(j)}\}$ and $\{r^{(j)}\}$, which satisfy the following relation:

$$\left\langle r^{(i)}, \hat{r}^{(j)} \right\rangle = 0, \qquad \text{for any } i \neq j.$$

In other words, they satisfy a bi-orthogonality relation. The bi-conjugate gradients (BiCG) algorithm may thus be derived from the Lanczos bi-orthogonalisation process. An implementation of BiCG can be realised as follows.

Algorithm:- BiCG
Input $x^{(0)}$; $r^{(0)} := f - Ax^{(0)}$, $p^{(0)} := r^{(0)}$,
$\hat{p}^{(0)} := \hat{r}^{(0)}$ where $\langle r^{(0)}, \hat{r}^{(0)} \rangle \neq 0$, $m := 0$;
For $m = 1, 2, \ldots$
Compute $Ap^{(m-1)}$; Compute $A^{\top}\hat{p}^{(m-1)}$;
$\alpha := \langle r^{(m-1)}, \hat{r}^{(m-1)} \rangle / \langle Ap^{(m-1)}, \hat{p}^{(m-1)} \rangle$;
$x^{(m)} := x^{(m-1)} + \alpha p^{(m-1)}$;

$r^{(m)} := r^{(m-1)} - \alpha A p^{(m-1)}$;
$\bar{\alpha} := \langle r^{(m-1)}, \hat{r}^{(m-1)} \rangle / \langle p^{(m-1)}, A^\top \hat{p}^{(m-1)} \rangle$;
$\hat{r}^{(m)} := \hat{r}^{(m-1)} - \bar{\alpha} \hat{p}^{(m-1)}$;
$\beta := \langle r^{(m)}, \hat{r}^{(m)} \rangle / \langle r^{(m-1)}, \hat{r}^{(m-1)} \rangle$;
$p^{(m)} := r^{(m)} + \beta p^{(m-1)}$;
$\hat{p}^{(m)} := \hat{r}^{(m)} + \beta \hat{p}^{(m-1)}$;
End-For;
End-Algorithm

Experience shows that the convergence history of the BiCG algorithm tends to be oscillatory. One way to smooth the oscillation is to use the BiCGSTAB algorithm, which is known as a smooth variant of the BiCG algorithm [17]. Another way to generate smooth convergence history is to use the GPBi-CG algorithm [19], which uses a generalized product-type method based on BiCG algorithm. Both algorithms avoid using the transpose of A in the BiCG procedure and gain faster and smoother convergence without increasing the computational cost. Both algorithms construct nonoptimal approximations in the same Krylov subspace as GMRES, but require less calculation efforts per iteration step compared with GMRES. Although the algorithms are not converging monotonically, their convergence rates are often faster than that of GMRES for cases of hard converging problems.

Theoretical and numerical tests to show the convergence behaviours of various Krylov subspace methods without using preconditioners can be found in the literature, e.g., [21]. In general, each algorithm has its advantages and disadvantages when applying to different problems depending on the properties of the coefficient matrix.

Preconditioning is essential to efficiently solve a linear system, especially for a nonsymmetric coefficient matrix such as the one in (4). A parallel preconditioner should be computed easily and be chosen in a way to suit parallel computation [22]. In the linear solver of the oil reservoir simulator (parallel Simbest-II), the additive Schwarz preconditioning, iterative algorithm preconditioning, and ILU preconditioning are combined together in order to achieve adaptive and efficient parallel preconditioners for each problem.

Let $A_{n\times n}$ be nonsingular, $P_{n\times m}$ be a projection operator, and $P^\top AP$ is invertible. (Here, coefficient matrix A is considered to be an $n \times n$ block, e.g., for equation (4), $n = 3$. P is considered to be a $n \times m$ block matrix, $m \leq n$.) It is possible to construct T_2 as

$$T_2 = P\left(P^\top AP\right)^{-1} P^\top,$$

and is straightforward to show the equality,

$$P^\top A T_2 = P^\top.$$

Suppose T_1 is an approximation of A^{-1}; one possible combined preconditioner following the idea in [12] may be defined as

$$B = T_1 + T_2(I - AT_1),$$

and it leads to a second equality,

$$P^\top AB = P^\top.$$

The above two equalities essentially point out the fact that $AT_2 = AB = I$. It is possible to use the above two equalities and mathematical induction in order to examine the effect of the combined preconditioner. The main result is summarised in the proposition below.

PROPOSITION. *Let B, T_1, and T_2 be defined as above. Suppose a given matrix P satisfies*

$$P^\top r^{(0)} = 0, \tag{5}$$

and the combined preconditioner B generates the residual vector sequence $\{r^{(n)}\}$, $n = 1, 2, \ldots$, and satisfies

$$r^{(n+1)} = \sum_{i=0}^{n} \alpha_i r^{(i)} + \sum_{i=0}^{n} \beta_i ABr^{(i)}, \tag{6}$$

where α_i and β_i are constants; then it can be proved that $r^{(k)}$ is constrained by

$$P^{\top} r^{(k)} = 0, \qquad \forall k \geq 1,$$

on the Krylov subspace.

The proposition suggests that the combined preconditioner has the residual $r^{(k)}$ being constrained, and its effect is to enforce orthogonality of P and $r^{(k)}$, and as a result the norm of the residual vector may be decreased monotonically. Similar properties were demonstrated by Aziz [11] and Walis [23]. Knowledge of P, $(P^{\top}AP)^{-1}$, and T_1 are needed in order to construct the preconditioner B.

First, the projection operator P may be chosen as

$$P = \begin{pmatrix} Q & & \\ & Q & \\ & & Q \end{pmatrix}, \tag{7}$$

where $Q^{\top} = (1, 1, \ldots, 1)_{Ng}$. There are other projection operators, which result to different preconditioners and require different linear solvers.

Second, the inverse matrix of $P^{\top}AP$ may be computed by means of either a direct method or an iterative method. A standard direct method is LU decomposition which solves a linear system $Ax = f$ by factorizing A into the form LU, where L is lower triangular, and U is upper triangular. Therefore, solution of $Ax = f$ reduces to a forward substitution and follows with a backward substitution. However, for discretised black oil model, it is not practical to compute and store these factors. An approximate factorisation (ILU) can be considered, say, $A \approx LU$ [24]. Therefore, the product LU can be used as a preconditioner in a Krylov subspace method for nonsymmetric problems. In this paper, the direct method is related to incomplete LU decomposition (ILU) and the iterative method is related to GMRES with ILU preconditioner. Comparisons of the two methods are given in Section 4.

Third, the construction of the preconditioner T_1 is based on ASP, which has its name rooted from alternating Schwarz method. For simplicity, the concept is briefly introduced here for the computational grid domain Ω which is split into two overlapped parts Ω_1 and Ω_2, such that $\Omega_1 \cap \Omega_2 \neq \emptyset$ and $\Omega_1 \cup \Omega_2 = \Omega$. Suppose the discretised black oil model in (4) has its field variables and the right-hand side vector being partitioned accordingly to $x = (x_{[1]}, x_{[12]}, x_{[2]})^{\top}$ and $f = (f_{[1]}, f_{[12]}, f_{[2]})$ with the subscripts in $[.]$ denoting the respective subdomains $\Omega_1 \setminus \Omega_2$, $\Omega_1 \cap \Omega_2$, and $\Omega_2 \setminus \Omega_1$. The rearranged system using the same notation $Ax = f$ takes the form

$$\begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_{[1]} \\ x_{[12]} \\ x_{[2]} \end{pmatrix} = \begin{pmatrix} f_{[1]} \\ f_{[12]} \\ f_{[2]} \end{pmatrix}.$$

Suppose R_s is a restriction operator which takes a vector defined on Ω and restricts it to Ω_s, and $R_s^{\top}$ is an extension operator which takes a vector defined on Ω_s and extends it with zeros to the rest of Ω. The rearranged set of linear equations may be written in the block Jacobi iteration form

$$A_{\{1\}} x_1^{(k)} = A_{\{1\}} x_1^{(k-1)} + R_1 \left(f - Ax^{(k-1)} \right), \qquad A_{\{2\}} x_2^{(k)} = A_{\{2\}} x_2^{(k-1)} + R_2 \left(f - Ax^{(k-1)} \right),$$

where

$$A_{\{1\}} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \qquad A_{\{2\}} = \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix},$$

and

$$x_{\{1\}} = \begin{pmatrix} x_{[1]} \\ x_{[12]} \end{pmatrix}, \qquad x_{\{2\}} = \begin{pmatrix} x_{[12]} \\ x_{[2]} \end{pmatrix},$$

and that using the block matrix notation as in (4) for $s = 1, 2$,

$$A_{\{s\}} = \begin{pmatrix} A_{s,11} & A_{s,12} & A_{s,13} \\ A_{s,21} & A_{s,22} & A_{s,23} \\ A_{s,31} & A_{s,32} & A_{s,33} \end{pmatrix}, \qquad x_{\{s\}} = \begin{pmatrix} x_{\{s\},1} \\ x_{\{s\},2} \\ x_{\{ss\},3} \end{pmatrix}.$$

Here $A_{s,11}$ is the coefficient submatrix obtained from pressure equation in oil phase, $A_{s,22}$ is obtained from the saturation equation in water phase, and $A_{s,33}$ is obtained from the saturation equation in oil (or gas) phase. The matrices $A_{s,ij}$ are heptadiagonal matrices. $x_{\{s\},1}$, $x_{\{s\},2}$, and $x_{\{s\},3}$ are the oil phase pressure, the water saturation, and the oil (or gas) saturation, respectively, in subdomain Ω_s. The block Jacobi iteration can be easily shown to give the algebraic form

$$x^{(k)} = x^{(k-1)} + \left(R_1^\top A_{\{1\}}^{-1} R_1 + R_2^\top A_{\{2\}}^{-1} R_2\right)\left(f - Ax^{(k-1)}\right).$$

Therefore, the block Jacobi iteration is equivalent to applying the preconditioner $R_1^\top A_{\{1\}}^{-1} R_1 + R_2^\top A_{\{2\}}^{-1} R_2$ to A. The idea can be easily extended to p subdomains, where $\Omega = \bigcup_{s=1}^{p} \Omega_s$ and not all subdomains have intersections, and it leads to the construction of T_1 as

$$T_1 = R_1^\top A_{\{1\}}^{-1} R_1 + \cdots + R_p^\top A_{\{p\}}^{-1} R_p. \tag{8}$$

The ASP must be used as right preconditioner in order to satisfy the relation in (6). The initial guess, $x^{(0)}$, which is a major factor of computational costs and convergence, needs to be selected to satisfy relation (5). As discussed in Section 2, the problem now is to solve the unknown quantities of the pressure and saturation at individual nodal points. For flows of reasonably slow flow rate, it is possible to consider the pressure distribution across the domain to be homogeneous. It is also possible to make use of these assumptions and the relation in (5) to obtain a good approximation. This will guarantee the norm of the iterative residual vector $r^{(n)}$ being converged to zero steadily and ensure B to be an efficient preconditioner.

Since parallel processing was used in the implementation, the ILU preconditioner was used only within local processors. In essence, both $P^\top AP$ and T_1 were implemented by using the additive Schwarz method as described above.

A comparison of the convergence behaviours of different Krylov subspace methods with ILU preconditioning of different filling-in levels for linear systems arising from partial differential equations can be found in [25], and the goal was to exploit the steadiness of convergence behaviour and the convergence rate of an algorithm for the solution of linear systems. These properties are summarised as follows. BiCGSTAB is one of the most effective algorithms; the convergence rate is faster than that of Orthomin and GMRES on many cases. However, BiCGSTAB shows an oscillatory convergence history while Orthomin and GMRES show monotonic convergence history. The convergence behaviour of GPBi-CG is more stable than that of BiCGSTAB, and the convergence rates of both algorithms are similar.

In general, preconditioning takes an important role in a Krylov subspace method. A good preconditioner not only can save much computational time, but allows linear systems to converge smoothly. In other words, preconditioning improves the stability of an algorithm. Numerical experiments are used to demonstrate the effect of preconditioning in terms of computational time and smooth convergence.

4. NUMERICAL EXPERIMENTS AND COMPARISON

In this section, some numerical experiments are used to demonstrate the relative efficiency of various Krylov subspace methods with two preconditioners. There are four check points in these numerical experiments.

First, the convergence behaviour of Orthomin, GMRES, BiCGSTAB, and GPBi-CG is examined by applying these algorithms to the linear solver of the oil reservoir simulator [15] with the same preconditioner. Second, the effect of two different preconditioners on the same Krylov subspace method is examined by solving the same oil field problem. Third, in order to obtain an effective linear solver for the oil reservoir simulation, different Krylov subspace methods with different preconditioners are applied to solve the same given problem. Fourth, the effect of the number of CPU on the performance of the parallelised linear solver is examined.

There are five different model test cases used in the numerical experiments. Test Case 1 is a standard black oil system, with an $80 \times 60 \times 2$ grid system, 109 wells, and the marching history is 8095 days. In this test case, there are 30 faults which give rise to 294 abnormal grid connections to form all the off-diagonal entries in the coefficient matrix A. Test Case 2 is a problem from the Society of Petroleum Engineers Solution Comparison Project on black oil simulators [26], with a $24 \times 25 \times 15$ grid system, 26 wells, and the marching history is 900 days. Test Case 3 is a simulation problem from an oil field in China [27] with a $35 \times 44 \times 34$ grid system, 36 wells, and the marching history is $4,962$ days. Test Case 4 is a practical black oil simulation problem [12] with a $92 \times 52 \times 18$ grid system, 15 rock types, and 51 wells, and a marching history of $16,436$ days. The phenomenon of crossflow occurs in this case, where fluid flow between layers through the wellpores, with a total of up to $1,552$ grids being involved in the crossflow. Test Case 5 is a full-scale dual porosity miscible flooding problem with 159,900 active grids ($130 \times 123 \times 10$), 377 wells, and the simulation history is 31.5 years. In this case, there are five different rock types, and the numbers of datum regions and equilibrium regions are 10 and 1, respectively [15]. Note that Cases 1 and 2 are smaller size and standard type of problems as compared to the other practical problems from industry.

The notation PRE-ILU refers to the combined preconditioner B, in which ILU decomposition is used to generate $(P^\top AP)^{-1}$ and, consequently, T_2. Similarly, PRE-ITER refers to the B which uses the GMRES algorithm to generate $(P^\top AP)^{-1}$, and therefore get T_2.

Algorithm:- Preconditioned (PRE-ITER) BiCGSTAB

Input $x^{(0)}$; $r^{(0)} := f - Ax^{(0)}$;
PRE-ITER:- Solve $Bv^{(0)} = r^{(0)}$ using GMRES for the construction of T_2;
$p^{(0)} := 0$; $m := 0$;
For $m = 1, 2, \ldots$
    BiCGSTAB to update $x^{(m)}$, $r^{(m)}$;
    PRE-ITER:- Solve $Bv^{(m)} = r^{(m)}$ using GMRES for T_2;
    Update $p^{(m)}$ according to BiCGSTAB;
End-For;
End-Algorithm

Note that the BiCGSTAB in the above algorithm may be replaced by a Krylov subspace method and that the convergence criteria [12] used in the test cases are independent of the choice of Krylov subspace methods and preconditioners. The outer Newton's iteration loop has the stopping criterion of $\|r^{(n)}\|_2 \leq 10^{-7}\|r^{(0)}\|_2$. The inner iteration loop for the linearised system returns to the Newton's loop when the criteria $\|x_1^{(n)} - x_1^{(n-1)}\|_\infty \leq \epsilon_1$ and $\|x_k^{(n)} - x_k^{(n-1)}\|_\infty \leq \epsilon_2$, $k = 1, 2$, are satisfied. Here x_k, $k = 1, 2, 3$, are the unknowns as in (4), and $\epsilon_1 = 0.0005$ and $\epsilon_2 = 0.00001$ are default criteria provided by the software.

For Orthomin and GMRES, the number of orthogonal vectors, m, is chosen to be 10 as default. Experiences show that $m \in [8, 12]$ with the combined preconditioner B gives the optimal results in most cases. In order to maintain the robustness of these algorithms, restarted GMRES(m),

restarted BiCGSTAB, and restarted GPBi-CG are considered. Restarted BiCGSTAB and GPBi-CG would avoid the behaviour of convergence stagnancy and divergence.

Different numbers of maximum number of iterations are used for the various iterative methods in order to maintain the consistent convergence control condition, and are chosen to be 25, 80, 720, and 720 for Orthomin, GMRES, BiCGSTAB, and GPBi-CG, respectively.

Both m and the maximum number of inner iterations may be altered to investigate convergence and performance properties of the above Krylov methods with preconditioners. Many realistic experiments are tested, and the results show that those choices can save much overall CPU timing and reduce the frequency of nonconvergence phenomena.

Tables 1–5 show the timings of sample runs of the five test cases using the methods and preconditoners as stated. The preconditioner PRE-ILU shows more favourable timings with the use of Orthomin and GMRES for Test Cases 1 and 2, as shown in Tables 1 and 2, while BiCGSTAB is becoming more attractive for Test Cases 3–5, as shown in Tables 3–5. Note that two CPUs were used to obtain the results as shown in Tables 1 and 2 because the test cases are relatively smaller to those in Tables 3–5. This suggests that Orthomin and GMRES are more suitable for well-defined and standard test problems of small sizes. BiCGSTAB with preconditioner PRE-ILU is suitable for larger industrial problems with many faults and wellpores as well as several different rock types, which all contribute to the slow convergence of the resulting linearised matrix. Table 5 shows significant different timings between Orthomin and GMRES when PRE-ILU is used. The maximum number of inner iterations being used in the tests are 25 and 80 for Orthomin and GMRES, respectively. Therefore, for very complex problems with very slow convergence in the linearisation process, it is often easy to reach the preassigned maximum number of inner iterations. This has typically happened in Test Case 5. For less complex problems, where computational work involved in a linearisation process is much less, GMRES or Orthomin combined with PRE-ILU shows similar convergence behaviour except for Test Case 5.

Table 1. Parallel linear solver (two CPU) elapsed time cost (sec.) of Test Case 1. (Linux PC cluster, P-III 450 Mhz, 100 Mb Ethernet.)

	Orthomin	GMRES	BiCGSTAB	GPBi-CG
PRE-ILU	776.92	714.45	1017.76	989.09
PRE-ITER	1135.21	621.98	67993.19	6852.05

Table 2. Parallel linear solver (two CPU) elapsed time cost (sec.) of Test Case 2. (Linux PC cluster, P-III 450 Mhz, 100 Mb Ethernet.)

	Orthomin	GMRES	BiCGSTAB	GPBi-CG
PRE-ILU	98.59	94.04	107.42	109.37
PRE-ITER	93.17	90.99	742.86	192.10

Table 3. Parallel linear solver (four CPU) elapsed time cost (sec.) of Test Case 3. (Linux PC cluster, P-III 450 Mhz, 100 Mb Ethernet.)

	Orthomin	GMRES	BiCGSTAB	GPBi-CG
PRE-ILU	4355.71	4361.71	2556.65	3000.95
PRE-ITER	4277.52	2121.57	3217.45	2508.25

Table 4. Parallel linear solver (four CPU) elapsed time cost (sec.) of Test Case 4. (Linux PC Cluster, P-III 450 Mhz, 100 Mb Ethernet.)

	Orthomin	GMRES	BiCGSTAB	GPBi-CG
PRE-ILU	10460.98	12988.04	8266.31	8231.44
PRE-ITER	11330.64	9695.36	202024.92	56628.65

Table 5. Parallel linear solver (four CPU) elapsed time cost (sec.) of Test Case 5. (Linux PC cluster, P-III 450 Mhz, 100 Mb Ethernet.)

	Orthomin	GMRES	BiCGSTAB	GPBi-CG
PRE-ILU	13606.87	29684.97	11811.6	10975.3
PRE-ITER	15676.58	14287.70	—	—

By choosing the preconditioner PRE-ITER, it is observed that GMRES has the advantage of shorter computational time compare with the other algorithms while BiCGSTAB and GPBi-CG have lost their privilege.

Iteration history and iteration statistics for Test Case 1 are shown in Figures 1–4. Numerous simulations, using different values of m, maximum numbers of inner iterations, and restart conditions, have been performed for Test Case 1. However, only a few of them were selected to be shown in these figures. The selection is to highlight possible different convergence histories during the simulations, which are denoted as Sim 1, 2, etc. in the figures.

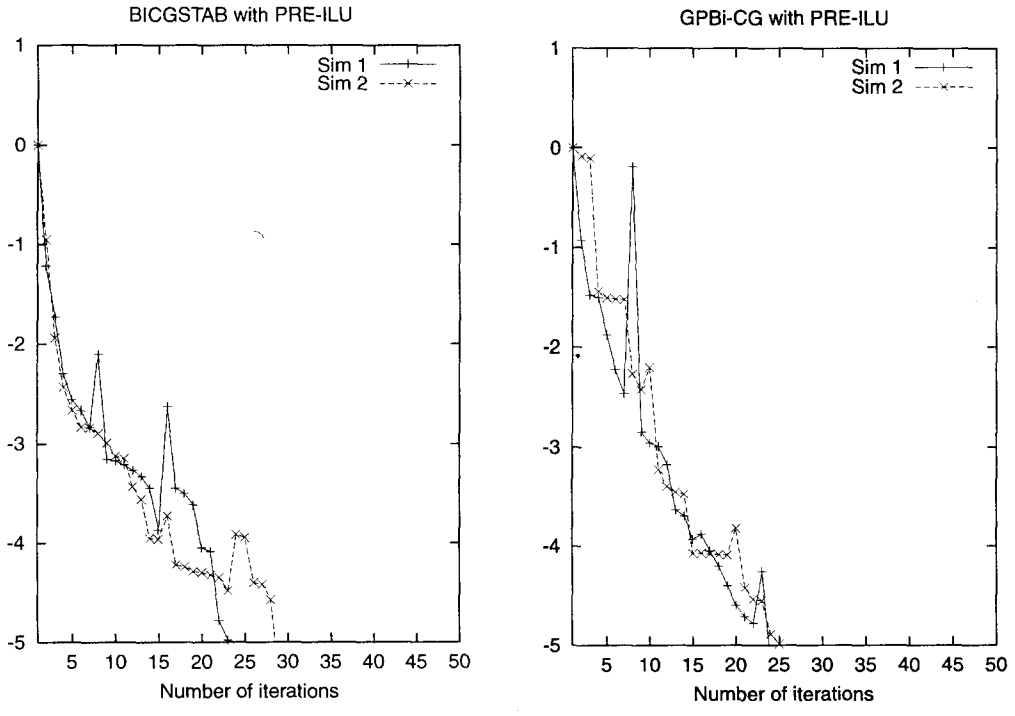

Figure 1. Iteration history of Krylov methods with PRE-ILU preconditioning.

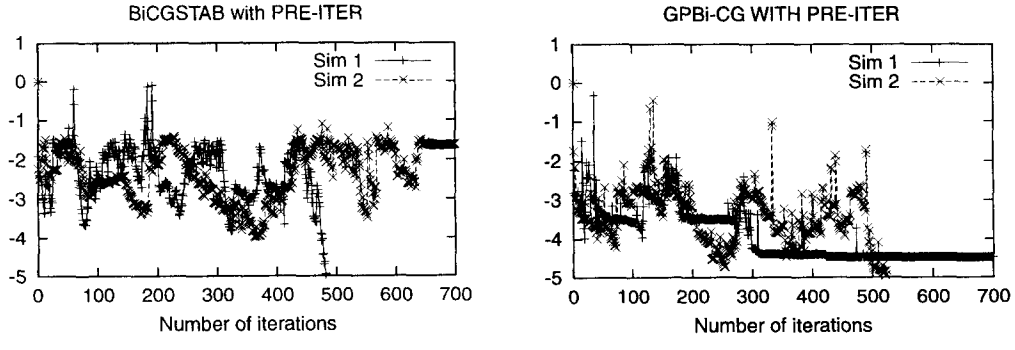

Figure 2. Iteration history of Krylov methods with PRE-ITER preconditioning.

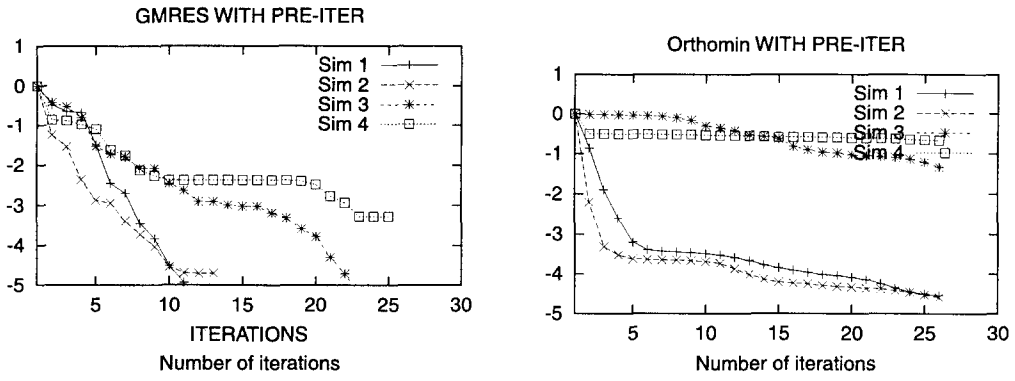

Figure 2. (cont.)

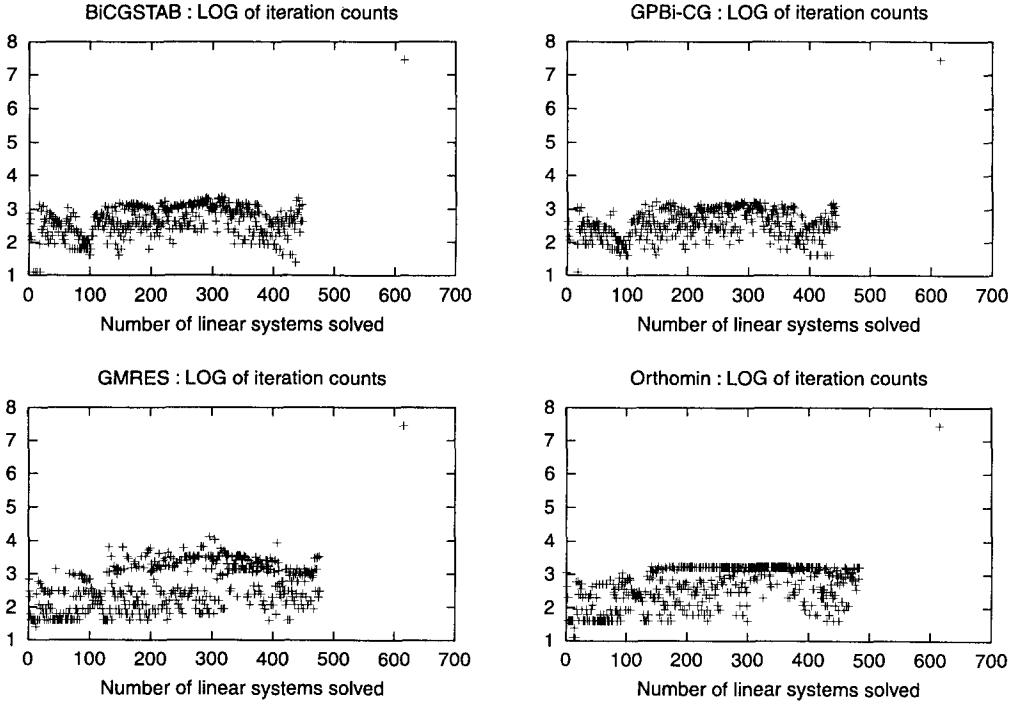

Figure 3. Iteration statistics of Krylov methods with PRE-ILU preconditioning.

Figure 1 shows the logarithmic plots of the iteration history of L_2-norm of the residual vector series $r^{(m)}$ for two Krylov subspace methods with the preconditioner PRE-ILU. The convergence rates for BiCGSTAB and for GPBi-CG are similar. Figure 2 shows the logarithmic plots of the iteration history of the L_2-norm of the residual vector series $r^{(m)}$ for four Krylov subspace methods with the preconditoner PRE-ITER. Orthomin and GMRES descend smoothly in general, but in some computer simulations as depicted in Figure 2, convergence seems to be stagnated during the iterative process.

Figures 3 and 4 show the iteration statistics, in which the number of linear systems being solved, as outer iteration progresses, is plotted against the corresponding logarithmic iteration counts. In the algorithms of BiCGSTAB and GPBi-CG, every iteration step includes two matrix multiplications, each of which forms the main part of the total computational cost. Thus, the total number of iteration steps serves to indicate the total computational costs of this algorithm. One can easily observe the huge amount of work required by the two preconditioners. From the

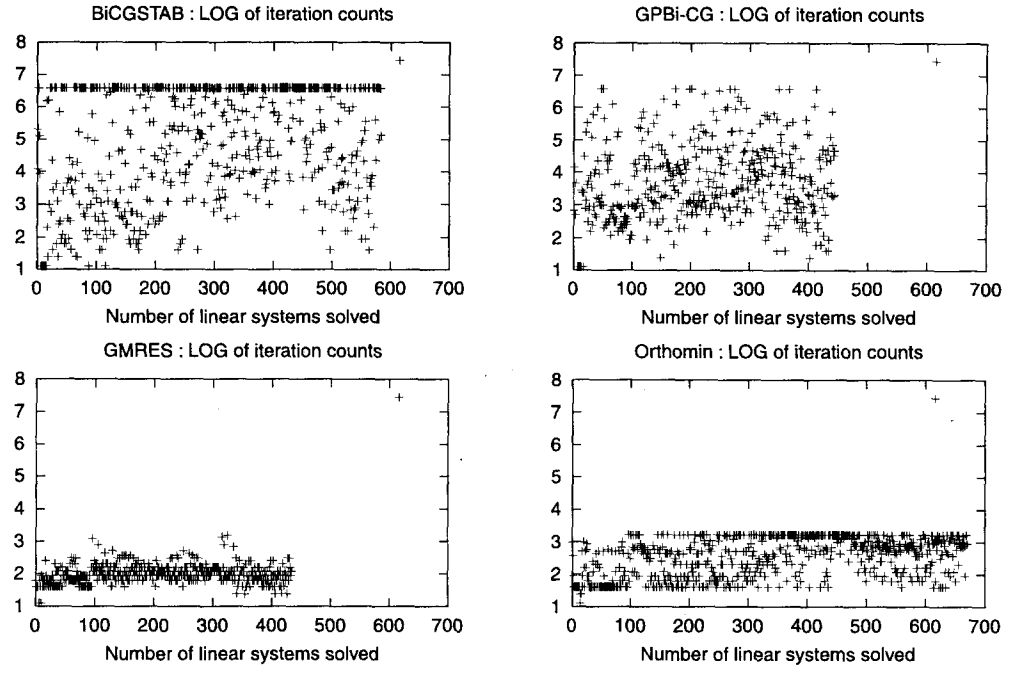

Figure 4. Iteration statistics of Krylov methods with PRE-ITER preconditioning.

iteration statistics shown in Figure 3, BiCGSTAB and GPBi-CG show very little difference in the computational costs when the preconditioner PRE-ILU is being used. From the iteration statistics shown in Figure 4, GPBi-CG shows more favourable computational costs than BiCGSTAB when the preconditioner PRE-ITER is used. As far as numerical solutions of the black oil model are concerned, GPBi-CG is more stable compared with BiCGSTAB.

By using PRE-ITER as the preconditioner, Figure 4 shows that the number of linear system solves for both BiCGSTAB and Orthomin often exceeds their maximum inner iteration limits and have to be interrupted. By using the other preconditioner, such interruptions decrease dramatically. This can be observed as the horizontal clustering of data in Figure 4. From Figures 3 and 4, it is possible to draw the conclusion that PRE-ILU is a better preconditioner than PRE-ITER in terms of computational costs. It is certainly true that PRE-ITER is far from the optimal preconditioner for BiCGSTAB or GPBi-CG, which can also be confirmed from the highly oscillatory convergence curves as shown in Figure 2. On the other hand, when PRE-ILU is used as a preconditioner for BiCGSTAB or GPBi-CG, the convergence rates for both methods are similar and their convergence shows monotonicity, as shown in Figure 1.

According to the role of a "good" preconditioner as described in Section 3, PRE-ILU may be considered as a good preconditioner when it is used with BiCGSTAB or GPBi-CG, and PRE-ITER may be considered as a good preconditioner when it is used with GMRES. However, it is yet too early to draw a conclusion at this stage that PRE-ILU is better than PRE-ITER when each of them is used with Orthomin. When the PRE-ITER is used with Orthomin, the choice of optimal tuning parameters, i.e., the inner convergence criterion and the maximum number of inner iterations, is unclear to the authors for black oil model applications. For a large number of mesh points and complex physics, such as Test Case 5, theoretical optimal tuning parameters derived for mildly nonlinear problems are certainly not suitable. In the case for oil reservoir simulations, a large number of experiments is required in order to provide empirical parameters. The authors believe that PRE-ITER with optimal tuning parameters is a good candidate for Orthomin [28].

The parallel efficiency of a parallel code is largely determined by the ratio of local computations over interprocessor communications. In general, the parallel efficiency improves as more grid cells

are used in each processor so that the communication cost can be dominated by a large amount of computations and the parallel efficiency degrades as the number of grid cells associated with a processor decreases. Test Cases 4 and 5 are good candidates to demonstrate this general rule because of the intense computation involved. Table 6 shows the parallel linear solver statistics of Test Case 4, and Table 7 shows the statistics of Test Case 5. As shown in Tables 6 and 7, when the number of CPUs increases from two to four, the elapsed time for the solver is almost halved. When the number of CPUs increases from four to eight, the elapsed time for the solver is reduced to just over one half of the former CPU numbers. Scalability is being observed for a given problem size from two processors to eight processors. For Test Case 4, it is also observed that the ratio of communication to computation increases to beyond one third. The authors believe that such a ratio is large enough to degrade the total computational costs as the number of grid sizes further decreases. The authors have performed many tests using Test Cases 1 and 2, and in each test, the cost of communication exceeds one third of the cost of local computation when the number of CPUs exceeds four.

Table 6. Parallel linear solver (GPBi-CG with PRE-ILU) statistics of Test Case 4. (Linux PC cluster, P-III 450 Mhz, 100 Mb Ethernet.)

	Two CPUs	Four CPUs	Eight CPUs
Solver Elapsed Time (sec.)	15733.80	8231.44	4745.32
Solver Communication Cost	1265.76	1407.03	2123.73
Communication/Computation	0.08045	0.17093	0.44754

Table 7. Parallel linear solver (GMRES with PRE-ITER) statistics of Test Case 5. (Linux PC cluster, P-III 450 Mhz, 100 Mb Ethernet.)

	Two CPUs	Four CPUs	Eight CPUs
Solver Elapsed Time (sec.)	28421.3	14287.7	8277.1
Solver Communication Cost	1097.9	1579.2	2284.0
Communication/computation	0.03863	0.11053	0.27594

5. CONCLUSIONS

In general, GMRES and Orthomin with a proper preconditioning such as PRE-ITER are effective and efficient methods for the solution of the linearised system in the black oil model with an unclear number of faults and geological properties. However, BiCGSTAB and GPBi-CG with the preconditioning PRE-ILU become more effective for a large class of problems in oil field simulation.

For smaller size problems of more standard type of simulation, such as Test Case 1, either Orthomin or GMRES with PRE-ILU is a good candidate. For increasing size problems of very complex nature with many geological properties, such as those in Test Cases 4 and 5, either BiCGSTAB or GPBi-CG with PRE-ILU is a good candidate.

The convergence rates of GPBi-CG and BiCGSTAB are similar. The convergence behaviour of GMRES and Orthomin is smooth, while the convergence behaviour of GPBi-CG and BiCGSTAB is oscillatory.

The parallel efficiency of a parallel code is largely determined by the ratio of local computations over interprocessor communications. The best parallel efficiency is achieved with large numbers of grid cells, where the communication cost can be dominated by a large amount of computations, and the parallel efficiency degrades as the number of grid cells decreases.

REFERENCES

1. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, (1995).
2. C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Nat. Bur. Stand.* **4**, 255–282, (1950).

3. M. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Stand.* **49**, 409–436, (1952).
4. W. Arnoldi, The principles of minimized iterations in the solution of the matrix eigenvalue problem, *Quart. Appl. Math.* **9**, 17–29, (1951).
5. J. Reid, On the method of conjugate gradients for the solution of large sparse systems of linear equations, In *Large Sparse Sets of Linear Equations*, (Edited by J. Reid), pp. 231–254, Academic Press, London, (1971).
6. K. Jea and D. Young, Generalised conjugate-gradient acceleration of non-symmetrizable iterative methods, *Linear Algebra Appl.* **34**, 159–194, (1980).
7. Y. Saad and M.H. Schultz, Conjugate gradient-like algorithms for solving non-symmetric linear systems, *Mathematics of Computation* **44**, 417–424, (1985).
8. Y. Saad and M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM Journal on Scientific and Statistical Computing* **7**, 856–869, (1986).
9. S.F. Ashby *et al.*, A taxonomy for conjugate gradient methods, *SIAM J. Numer. Anal.* **27**, 1542–1568, (1990).
10. G.H. Golub and H.A. Van der Vorst, Closer to the solution: Iterative linear solvers inbook The State of the Art in Numerical Analysis, Clarendon Press, Oxford, (1997).
11. K. Aziz and A. Settari, *Petroleum Reservoir Simulation*, Applied Science, London, (1979).
12. *SimBest II Dual-Porosity, Black Oil Reservoir Simulator User's Manual, Simbest II, Version 2.0*, Released August 1989, Scientific Software-Intercomp, Inc.; See also `http://www.bakerhuges.com/bakeratlas/solutions/workbench/index.htm` for electric manual of Version 3.4.5. Released September 1999.
13. J. Abate, P. Wang and K. Sepehrnoori, *Parallel Compositional Reservoir Simulation on a Cluster of PCs*, The University of Texas at Austin, (1998).
14. S. Verdiere, L. Quettier, P. Samier and A. Thompson, Applications of a parallel simulator to industrial test cases, SPE 51887, (1999).
15. W. Liu, J. Cao, A. Mezzatesta and P. Zhu, Parallel reservoir simulation on shared and distributed memory system, SPE 64797, (2000).
16. P.K.W. Vinsome, Orthomin, An iterative method for solving sparse sets of simultaneous linear equations, SPE 5729, (1976).
17. H.A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems, *SIAM Journal on Scientific and Statistical Computing* **12**, 631–644, (1992).
18. G.L.G. Sleijpen and H.A. Van der Vorst, *Computational Fluid Dynamics Review 1995*, (Edited by M. Hafez and K. Oshima), pp. 457–476, ISBN 0 471 95589 2, Wiley, Chichester, (1995).
19. S.-L. Zhang, GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, *SIAM J. Sci. Comput.* **18** (2), 537–551, (March 1997).
20. A. Greenbaum, *Iterative Methods for Solving Linear Systems, SIAM Frontiers in Applied Mathematics, Volume 17*, SIAM, Philadelphia, (1997).
21. N.M. Nachtigal, S.C. Reddy and L.N. Trefethen, How fast are nonsymmetric matrix iterations?, *SIAM J. Matrix Anal. Appl.* **13**, 778–795, (1992).
22. J. Sun and J. Cao, A class of local Green-like parallel preconditioner algorithm for elliptic discrete equations II: For non self-conjugate equations, *Mathematica Numerica Sinica* **18** (2), (1995).
23. J.R. Walis, Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration, SPE 12265, (1983).
24. R.S. Varga, Factorisation and normalised iterative methods, In *Boundary Problems in Differential Equation*, (Edited by R.E. Langer), pp. 121–142, The University of Wisconsin Press, Madison, WI, (1960).
25. J. Cao, Numerical experiments and analyses of Krylov subspace methods on one kind of PDE, *Journal on Numerical Methods and Computer Applications* **20** (4), (1999).
26. J. Killough, Ninth comparative solution project: A reexamination of black oil simulation, Presented at 13^{th} *SPE Symposium of Reservoir Simulation*, February 1995, San Antonio, TX; `http://www.spe.org`.
27. Finery oil reservoir numerical simulation of large-scale oil field, National 863 Project (2000); `http://www.rdcps.ac.cn`.
28. H.A. Van der Vorst and C. Vuik, GMRESR: A family of nested GMRES methods, *Num. Lin. Alg. with Appl.* **1**, 369–386, (1994).
29. D.E. Keyes, Y. Saad and D.G. Truhlar, *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, SIAM, Philadelphia, (1995).
30. P. Sonneveld, CGS, A fast Lanczos-type solver for nonsymmetric linear systems, *SIAM Journal on Scientific and Statistical Computing* **10** (1), 36–52, (1989).